

Low-Altitude Laser Altimeter to Assist UAV Autolanding

By

Copyright 2012

Nicholas Marshall Bergmann

Submitted to the graduate degree program in Electrical Engineering and the Graduate Faculty of  
the University of Kansas in partial fulfillment of the requirements for the degree of Master of  
Science

---

Chairperson, Dr. Christopher Allen

---

Committee Member, Dr. Shannon Blunt

---

Committee Member, Dr. Rongqing Hui

Date Defended: July 9<sup>th</sup>, 2012

The Thesis Committee for Nicholas Marshall Bergmann  
certifies that this is the approved version of the following thesis:

Low-Altitude Laser Altimeter to Assist UAV Autolanding

---

Chairperson, Chris Allen

Date Approved: July 9<sup>th</sup>, 2012

## Abstract

This project presents a low-altitude laser altimeter system to assist UAV autoland. This system generates aircraft altitude and attitude estimates; it consists of laser illuminators, a digital imaging unit (image sensor, lens, and optical filter), and a processor. The purpose of this project is to provide real-time altitude above terrain, roll, and pitch to an existing onboard autopilot system with sufficient accuracy to facilitate the automated landing of the UAV on uncharted landing strips.

The laser altimeter system presented by this project employs an optical triangulation concept to estimate the altitude above terrain, roll, and pitch of the aircraft. To implement this system, the Leopardboard 365 is used for the processor; the digital imaging unit is comprised of the Aptina MT9P031 CMOS image sensor, the SY110M ultra-wide angle lens, and the Edmund Optics 780-nm bandpass optical filter; and, four US Lasers 780-nm laser diode modules act as the laser illuminators. Finally, a test fixture was built to assess the speed and accuracy of the system.

System Parameters	Value
Altitude Operating Range	1 – 10 m
Roll Operating Range	$\pm 15^\circ$
Pitch Operating Range	$\pm 15^\circ$
Baseline Lengths	0.5 m
Grazing Angles	$57^\circ$
Operating Wavelength	780 nm
System Power Consumption	2.3 W

System Performance Metrics	Value
Altitude Error (Avg)	2.28%
Roll Error (Avg)	$1.00^\circ$
Pitch Error (Avg)	$1.13^\circ$
Output Data Rate (Avg)	0.98 Hz

Design Performance Metrics	Value
Altitude Error (Max)	5%
Roll Error (Max)	$3^\circ$
Pitch Error (Max)	$3^\circ$
Output Data Rate (Min)	10 Hz

## **Acknowledgements**

I would like to thank my advisor, Dr. Allen, for all his guidance and support; thank you for providing me with an opportunity to work on this complex and interesting project. I would also like to thank Amber Peterman; thank you for assisting with the construction of the test fixture and for all your encouragement.

## Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1. Background and Motivation	1
1.2. Design Requirements	2
1.3. Existing Technologies	2
<b>2. Methodology</b>	<b>5</b>
2.1. System Description and Implementation	5
2.1.1. Optical Triangulation Concept (One-Laser System)	5
2.1.2. Two-Laser System	7
2.1.3. Generating Altitude and Attitude Solutions using the Two-Laser System	9
2.1.4. Expansion to a Four-Laser System	10
2.1.5. Combining the Altitude Estimates	13
2.2. System Analysis and Design	14
2.2.1. Ensuring the Laser Beams Cross	14
2.2.2. Ensuring the Laser Beams are within the Field-of-View	16
2.2.3. Grazing Angle Selection	17
2.2.4. Baseline Lengths	19
2.2.5. Wavelength Selection	21
2.2.6. Maximum Laser Power (Eye-Safety)	22
2.2.7. Minimum Laser Power	22
2.2.8. Optical Filter Inclusion	23
2.2.9. System Simulation	23
<b>3. Presentation of Work</b>	<b>26</b>
3.1. Processor	26
3.1.1. Software	28
3.2. Image Sensor	29
3.3. Lens	29
3.4. Optical Filter and Laser Illuminators	29
<b>4. Test and Evaluation</b>	<b>31</b>
4.1. Test Fixture	31
4.2. Processor Speed	36

4.3. Accuracy.....	37
5. Conclusion .....	40
6. Future Work .....	41
7. References.....	43
8. Appendices .....	45
A. Mathematical Analysis .....	45
A.1. Optical Triangulation.....	45
A.2. One-Laser Altitude Above Terrain.....	45
A.3. Laser Offset from the Horizontal Plane of the Camera.....	45
A.4. Two- Laser Altitude Above Terrain Derivation .....	46
A.5. Maximum Laser Power (Eye-Safety).....	47
A.6. Minimum Laser Power .....	47
B. System Code (C).....	49

## **1. Introduction**

### **1.1. Background and Motivation**

Invented during WWI, unmanned aerial vehicles (UAVs) were originally designed for military use. Since then, advances in technology have allowed for the use of UAVs to expand into many other facets of society including scientific research, natural resource exploration, and commercial, industrial, and governmental surveillance. Like many automated systems, the use of UAVs has traditionally been reserved for activities that are too “dull, dirty, or dangerous” for manned aircraft. However, UAVs offer many significant advantages over manned aircraft including reduced initial and operating costs, ease of transport, versatile basing options, and the capability for use in hazardous environments or conditions. Despite these advantages, UAVs still have many barriers to overcome before their use becomes wide-spread. One of these barriers is the low degree of autonomy that the UAV has, specifically, the inability of the UAV to take-off and land without human involvement. Skilled UAV operators are required to launch and, more importantly, land UAVs. Because these skilled operators are at a premium, a system that could provide accurate, real-time aircraft altitude and attitude could significantly broaden the market application of UAVs by reducing the requisite operator skill level. Moreover, this system could be integrated in such a way that operator input would no longer be required and the UAV could land autonomously (autoland).

With this goal in mind, this project demonstrates the development of a low-altitude laser altimeter system that will provide precise altitude above terrain data to existing autopilot technology thereby enabling the autolanding of a UAV. This system will also develop aircraft attitude data which will be used to refine the altitude data and which will also be utilized by the aforementioned aircraft autopilot. The capabilities of this system, when coupled with

preprogrammed waypoint flight navigation, enable the operation of the UAV by aeronautical novices without the involvement of a skilled operator.

## **1.2. Design Requirements**

First and foremost, the autoland system is constrained in size and weight by the physical dimensions and flight characteristics of the UAV itself. While the autoland system is applicable to both large and small UAVs, the autoland system developed in this project was designed with the 40% scale YAK-54 aircraft as the candidate platform. The 40% YAK-54 is a \$14,000, 43-lb., radio-controlled aircraft with a 3.277-m wingspan, 2.91-m fuselage length, and 15-lb. maximum payload.

The low-altitude laser altimeter system is additionally constrained by the operational requirements of the UAV. For the system to be useful during the autoland process, it must have an output data update rate of approximately 10 Hz with minimal data latency. Because the system will only be used during the low-altitude operation of landing, the applicable altitude range for the system is 1 m to 10 m, where the minimum operating altitude of 1 m accounts for the distance from the ground the system will be when mounted to the UAV. The desired accuracy for the altitude outputs of the system is 5% across the applicable altitude range. The system should also be able to provide aircraft attitude (roll and pitch) within 3° of accuracy over a range of  $\pm 15^\circ$ .

## **1.3. Existing Technologies**

There are many other altimeter solutions that have the potential to satisfy the requirements of the proposed autoland system. The first candidate technology is GPS. GPS is used in many autopilot systems and provides an array of useful information including aircraft position, velocity, and altitude. However, due to the approximate 3-m uncertainty in the altitude provided



by GPS, GPS is more suited to in-flight use as it does not provide accurate enough data for the precise nature of autolandings.

The second candidate technology is the radar altimeter. The radar altimeter has many advantages including the ability to operate in all weather conditions and immunity to smoke, fog, haze, etc. The radar altimeter, however, is susceptible to jamming, inaccurate when flying over low-reflectivity surfaces, and comparatively heavy. Roke Manor Research Ltd. produces the Miniature Radar Altimeter Type 2 [8]; this system has the following specifications: 77-GHz operating frequency, 10-Hz update rate, and 2-cm accuracy for altitudes ranging from 20 cm to 100 m. The Roke Manor Research Ltd. system costs \$17,000, consumes 3 W, weighs 14.1 oz., and measures 5.5" by 3" by 1.8".

The third candidate technology is the lidar. Lidars have the required accuracy; however, their output is corrupted by unknown aircraft attitude unless the beam is either gimbaled or scanned. Laser Technology, Inc. produces a non-scanning lidar system, the Universal Laser Sensor (ULS) [15]; this system has the following specifications: 905-nm wavelength and 2-cm accuracy for altitudes ranging from 46 cm to 500 m. The Laser Technology Inc. system costs \$1,800, consumes 2 W, weighs 28.2 oz., and measures 5.3" by 4.7" by 2.5".

The fourth candidate technology is the acoustic sensor. Acoustic sensors are accurate and have a small form factor. Yet, their performance is degraded by noise sources including wind, turbulence, vibration, and aircraft engines. Additionally, they can interfere with other aircraft systems through electromagnetic (EM) and/or radio frequency (RF) coupling. Devantech produces an acoustic sensor, the SRF08 High Performance Ultrasonic Range Finder [13]; this system has the following specifications: 40-kHz operating frequency and 3-cm accuracy for altitudes ranging from 3 cm to 6 m. The Devantech sensor costs \$65, consumes 0.1 W, weighs

0.4 oz., and measures 1.7” by 0.8” by 0.7”. Devantech does not provide specifications for the cost, power, weight or size of the external interfacing and processing systems.

The final candidate technology is the infrared range sensor. The most notable attribute concerning infrared range sensors are their significantly limited working distances.

HeliCommand produces the Profi-series [2] [12] platform stabilizer which uses a combination of four optical imaging systems, a 3-axis accelerometer, gyros, and a barometric altimeter. The barometric altimeter provides altitude data up to altitudes of 30 m (no accuracy given). An optional infrared range sensor can be included in the system to provide 10-cm altitude accuracy up to altitudes of 1.5 m. The system costs \$4,500, consumes 1 W, weighs 8.1 oz., and has a height of 2.9” and a diameter of 2.9”.

## 2. Methodology

### 2.1. System Description and Implementation

#### 2.1.1. Optical Triangulation Concept (One-Laser System)

To provide accurate altitude above terrain data for the low-altitude laser altimeter proposed by this project, an optical triangulation concept was developed that utilizes a laser as the illuminator and a one-dimensional line camera as the receiver. When arranged in the geometry depicted by Figure 1, the angle-of-arrival,  $\theta$ , of the scattered light from the laser at the line camera can be related to the distance from the scattering surface,  $D$ , by Equation (1); where  $x$  is the distance past the center of the camera at which the laser strikes the scattering surface,  $f$  is the focal length of the lens, and  $u$  is the distance between the center of the camera and the illumination point of the scattered light in the camera.

$$xf = uD \quad (1)$$

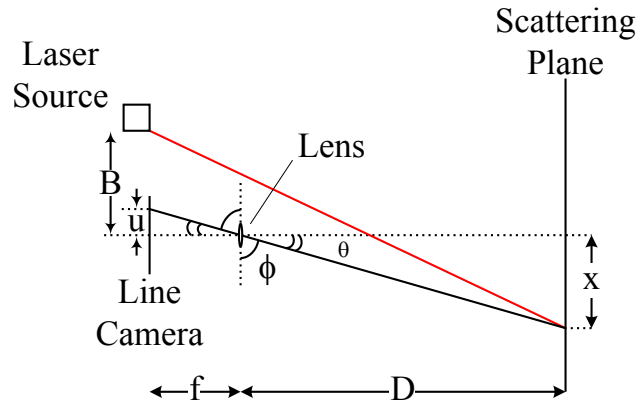


Figure 1 – Optical triangulation concept

The distance  $u$  can be determined by detecting the most significantly illuminated pixel in the line camera. However, as the line camera is a one-dimensional array of discrete pixels, the distance  $u$  will be quantized by the physical size of each pixel. Similarly, the value for the angle-of-arrival will also be quantized into discrete values. It should also be noted that the relationship

expressed by Equation (1) is directly dependent upon the focal length,  $f$ , of the lens used to focus the line camera.

Using the relationship in Equation (1), for a known baseline length,  $B$ , and laser illumination angle,  $\phi_1$ , the altitude above terrain,  $h$ , can be determined using Equation (2), where the laser source is located at  $P_1$  and the camera is located at  $P_0$  as depicted in Figure 2.

$$h = \frac{B}{\tan \theta_1 - \tan \theta_0} \quad (2)$$

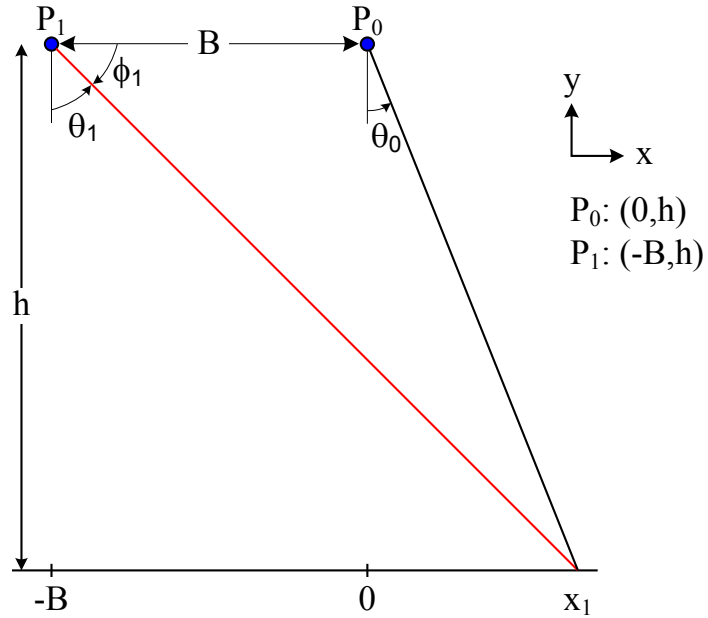


Figure 2 – One-laser system configuration

Figure 2 assumes that the line camera and laser source are located on the same horizontal plane. If, however,  $P_1$  and  $P_0$  do not lie in the same plane, as depicted in Figure 3, the laser source can be projected onto the plane of the line camera using the relationship found in Equation (3).

$$B' = B - a \cot \phi_1 \quad (3)$$

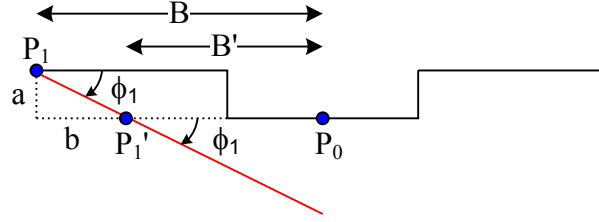


Figure 3 – Laser illuminator vertically offset from line camera

The one-laser system depicted by Figure 2 could be mounted to the underside of a UAV to provide an altitude estimate during landing. However, this one-laser system is applicable only to scenarios where the platform is level with the terrain. This means that the aircraft must be parallel to the landing surface and be experiencing zero roll and zero pitch. When this scenario is not satisfied,  $\theta_l$  from Figure 2 becomes unknown, as  $\theta_l$  and  $\phi_l$  are no longer complementary angles. Thus, variations in the platform attitude result in an unreliable and inaccurate altitude above terrain estimates.

#### 2.1.2. Two-Laser System

Cross-winds and sloping landing strips are just two of the many scenarios in which aircraft roll and pitch are non-zero during landing. To resolve the issues presented by aircraft attitude variations, a second laser illuminator was added to the one-laser system; this two-laser system is depicted in Figure 4. The addition of the second laser illuminator provides the system with two data points which are used to resolve two unknowns, the variation in the platform attitude (roll or pitch),  $\beta$ , and the altitude above terrain,  $h$ .

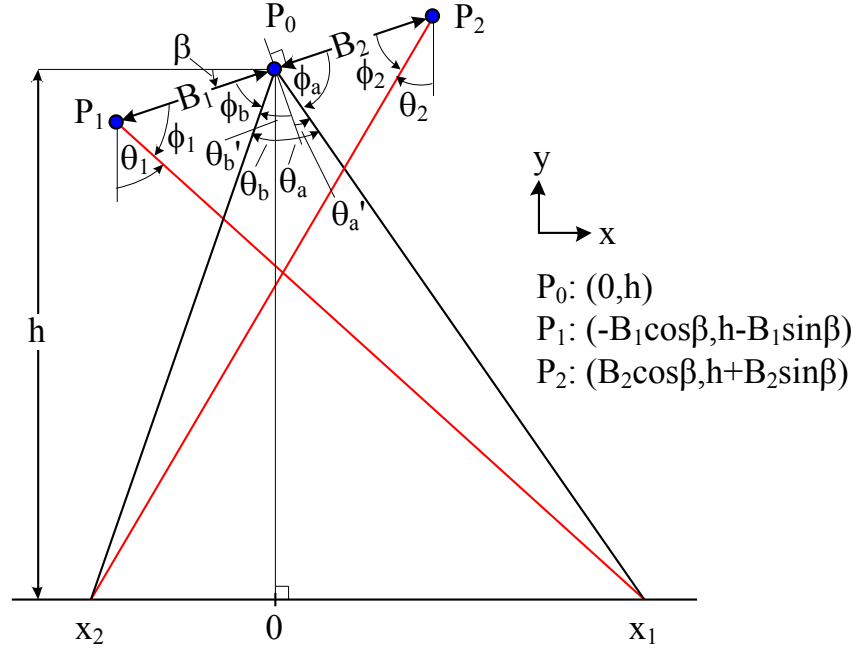


Figure 4 – Two-laser system configuration

Equations (4) and (5) are closed-form solutions for the altitude above terrain,  $h$  from Figure 4, where  $B_1$ ,  $B_2$ ,  $\phi_1$ , and  $\phi_2$  are known quantities, and  $\phi_a$  and  $\phi_b$  can be determined by detecting the most significantly illuminated pixels of the line camera. Since Equations (4) and (5) both calculate the same altitude above terrain, the expressions for  $h_1$  and  $h_2$  can be equated.

Traditionally, this would produce a closed-form expression for  $\beta$ ; however, due to the complex nature of these equations, a closed-form solution for  $\beta$  is not practical. Although an explicit expression for  $\beta$  has not been developed, solutions for  $\beta$  can be found by searching for the value of  $\beta$  that minimizes the difference between  $h_1$  and  $h_2$ . Given  $\phi_a$  and  $\phi_b$ ,  $h_1$  and  $h_2$  can be found for all feasible values of  $\beta$ . The  $\beta$  that minimizes the difference between  $h_1$  and  $h_2$  can be used as an approximation of the actual  $\beta$  value. The estimate of  $\beta$  is then a function of the error inherent in  $\phi_a$  and  $\phi_b$  as well as the granularity of the  $\beta$  search. The altitude estimates,  $h_1$  and  $h_2$ , can then be averaged to produce the final altitude estimate.

$$h_1 = \frac{B_1 \sin \beta \cot(\beta - \varphi_1) - B_1 \cos \beta}{\cot(\beta - \varphi_1) - \cot(\beta - \varphi_a)} \quad (4)$$

$$h_2 = \frac{B_2 \sin \beta \cot(\beta + \varphi_2) - B_2 \cos \beta}{\cot(\beta + \varphi_b) - \cot(\beta + \varphi_2)} \quad (5)$$

Figure 4 depicts a system in which two laser illuminators are located on opposite sides of the camera and are aligned in the same horizontal plane as the field view of the camera. It is envisioned that this configuration would facilitate the mounting of the line camera in the center of the underside of the fuselage of the aircraft and the laser illuminators would be mounted on the underside of the wings of the aircraft. While it is understood that rarely do the underside of both the fuselage and the wings lie in the same horizontal plane, for simplicity, henceforth this condition will be assumed. To accommodate for positive or negative vertical offsets of the laser illuminators relative to the line camera, see Equation (3).

### 2.1.3. **Generating Altitude and Attitude Solutions using the Two-Laser System**

At least two methods exist for a practical system to generate real-time aircraft attitude and altitude estimates. The first method involves constructing a look-up table (LUT) whose cells contain the altitude above terrain,  $h$ , and the aircraft attitude,  $\beta$ . Using the physical layout of the pixels in a line camera as an one-dimensional array, the index numbers of the two most significantly illuminated pixels (one pixel corresponding to each laser illuminator) could be used to address the cells of the LUT; where one index is used to specify the row and the other used to specify the column of the LUT. The values populating the LUT could be determined analytically by passing system specific parameters to a numerical simulator, such as Matlab. Likewise, the values could also be determined experimentally by a calibration process after the system has been installed on the target platform. The LUT method involves less real-time processing and is an attractive option when the system has enough memory to accommodate the LUT.

The second method for generating  $h$  and  $\beta$  involves implementing the  $\beta$  search approach described in Section 2.1.2. This method involves more real-time processing and is an attractive option when the system has adequate processing capabilities. This project employs the  $\beta$  search approach; thus, aspects of the analysis that follows will not be applicable to the LUT method. The justifications for choosing the  $\beta$  search approach over the LUT method can be found throughout Section 3.

#### **2.1.4. Expansion to a Four-Laser System**

Aircraft attitude is comprised of three independent parameters, roll, pitch, and yaw. In reference to this laser altimeter system, yaw can be ignored as changes in yaw do not affect the plane of the airframe relative to the ground but rather only the heading of the airframe relative to the velocity vector. Roll and pitch, on the other hand, can significantly degrade the altitude above terrain estimate. The two-laser system can resolve either the roll or pitch of the aircraft to refine the altitude estimate. However, if the aircraft is experiencing both roll and pitch simultaneously, the two-laser system will prove inadequate in accurately resolving the aircraft altitude. Thus, the low-altitude laser altimeter system was expanded to include two orthogonally aligned two-laser systems thereby comprising a four-laser system.

Each of the two-laser systems used in the four-laser system will resolve one of the unknown aircraft attitude parameters in addition to providing an altitude estimate. The altitude estimate generated by each of these two-laser systems will be degraded by the out-of-plane angle; however, utilizing the resolved roll and pitch values, these altitude estimates can be combined to remove these degradations. This final altitude estimate is then refined of all aircraft attitude effects.



Figure 5 and Figure 6 illustrate the four-laser system configuration when mounted on the underside of the fuselage of the target aircraft. In this configuration, one of the two-laser systems is aligned with the fuselage axis of the UAV and the other is aligned orthogonally along the UAV wing axis. In the four-laser system, the two orthogonally aligned line cameras could be replaced by a single two-dimensional image sensor. Ideally, in this arrangement, each of the two-laser systems would illuminate pixels within a single row and column, respectively, of the 2D image sensor. Figure 5 demonstrates this arrangement; where the red lines depict laser illumination, the shaded gray areas depicts the horizontal and vertical fields-of-view of the image sensor, and the intersecting arrays of boxes depict the projection of the row and column of pixels from the image sensor onto the landing surface. For Figure 6, the red lines depict laser illumination and the dashed lines depict the field-of-view of the camera.

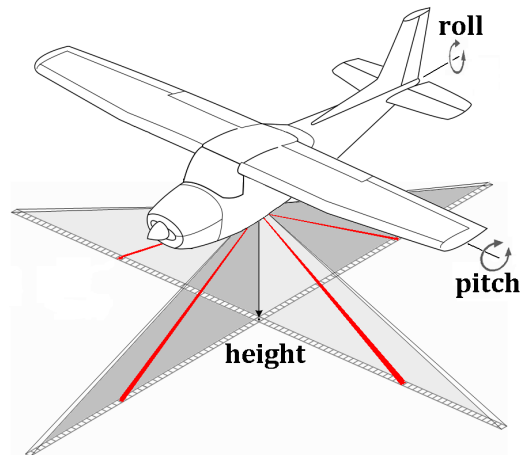


Figure 5 – Four-laser system configuration

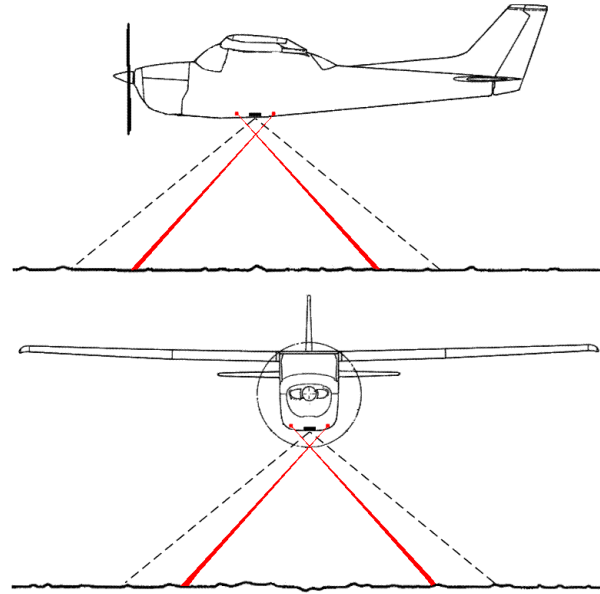


Figure 6 – Side view (top) and forward view (bottom) of the four-laser system

Using a 2D image sensor as opposed to two line cameras has numerous benefits. Foremost, it relaxes the requirement that the lasers and the FOV of the line camera lie in the same plane. When using the 2D image sensor, the lasers should be aligned with a single pixel row or pixel column of the image sensor. If, however, this alignment cannot be achieved, the process used to search for the most significantly illuminated pixels must be updated to search across multiple pixel rows/columns.

Additionally, the use of a 2D image sensor can allow for the reduction in the number of lasers or the extraction of additional information. Due to the fact that there are three unknowns in the system, height, roll, and pitch, the number of lasers could be reduced from four to three. Three lasers would prove adequate in solving for the three unknowns of the system; however, the equations used to solve for these unknowns and the method by which the altitude estimate is refined would require corresponding adaptations. To extract additional information, the number of lasers could be expanded. The use of four or more lasers would allow for the extraction of curvature and slope of the local terrain. This project employs a four-laser system for which

estimates of the aircraft altitude, roll, and pitch are generated. The justifications for this choice can be found throughout Section 3.

#### 2.1.5. Combining the Altitude Estimates

As previously mentioned, using the four-laser system, the altitude estimates generated by each of the two-laser systems must be combined to create an altitude estimate that is refined of all the effects of aircraft attitude. The two-laser system aligned with the fuselage of the UAV will output  $\beta_p$ , the aircraft pitch, and  $h_p$ , an altitude estimate that has been refined of aircraft pitch. The two-laser system aligned with the wings of the UAV will output  $\beta_r$ , the aircraft roll, and  $h_r$ , an altitude estimate that has been refined of aircraft roll. Each of the altitude estimates can be refined of the out-of-plane angle using Equations (6) and (7). These two altitude estimates,  $h'$  and  $h''$ , which have been refined of both aircraft pitch and roll, can then be averaged to create the final altitude estimate of the four-laser system, as shown in Equation (8).

$$h' = h_r \cos \beta_p \quad (6)$$

$$h'' = h_p \cos \beta_r \quad (7)$$

$$h = \frac{h' + h''}{2} \quad (8)$$

Equations (6) and (7) require each of the altitude estimates,  $h_r$  and  $h_p$ , to be independent of each other. Moreover, the roll and pitch values,  $\beta_r$  and  $\beta_p$ , produced by the system must likewise be independent. Intuitively, all of these parameters appear to be independent as the two-laser systems are mounted orthogonal to each other and the entire system experiences aircraft rotations equally. However, a more detailed analysis ensures this independence.

Aircraft attitude can be represented using Euler angles. The coordinate system of an aircraft with non-zero attitude (roll, pitch, and/or yaw) can be represented by a series of rotations from the frame of reference (an aircraft with zero roll, pitch, and yaw). Using Figure 7 for reference, it

can be seen that aircraft roll rotates the y-z plane about the x-axis. Similarly, aircraft pitch rotates the x-z plane about the y-axis. Each of these operations corrupt the altitude estimates yet have no effect on the out-of-plane axis. Equations (6) and (7) verify this fact. When solving for aircraft pitch and altitude, a projection onto the roll plane is created; when solving for aircraft roll and altitude, a projection onto the pitch plane is created. Thus, each of the altitude estimates need to be adjusted for the out-of-plane angle, but, the roll and pitch measurements themselves will be accurate.

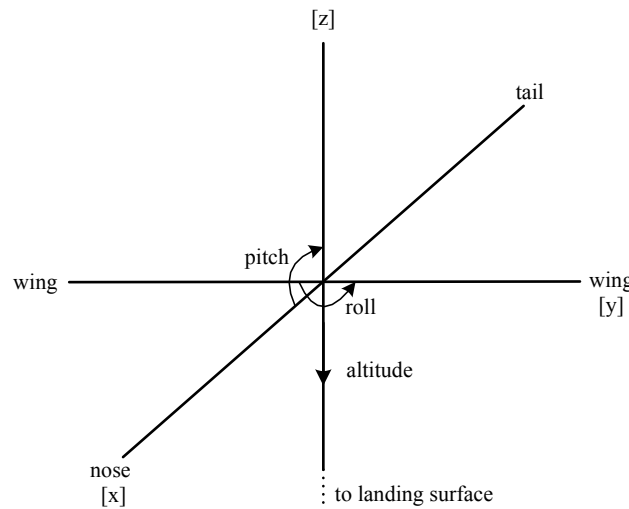


Figure 7 – Aircraft orientation and axes

## 2.2. System Analysis and Design

### 2.2.1. Ensuring the Laser Beams Cross

The optical triangulation concept that the two-laser system, and subsequently the four-laser system, is built upon requires that the beam of each of the laser illuminators cross the centerline of the image sensor prior to reaching the landing surface. Ensuring the laser beams cross the centerline of the image sensor also ensures that the beams will either cross over each other before they reach the ground or be collocated when they reach the ground; this is critical in preventing the illuminated pixels of the image sensor from becoming ambiguous. The worst-case altitude

with respect to ensuring the beams cross is 1 m; as this is the minimum operating altitude of the system, it presents the least amount of horizontal distance the beams will travel prior to striking the ground. Table 1 shows the upper bounds on the grazing angles,  $\phi_1$  and  $\phi_2$  from Figure 4, for a 1-m altitude above terrain and aircraft attitudes ranging from 0° to 30° (the aircraft attitude range was expanded beyond normal operating ranges to demonstrate the pattern associated with this analysis).

Table 1 – Maximum grazing angles to ensure beams cross

Aircraft Attitude	Grazing Angle
0°	63°
5°	63°
10°	63°
15°	64°
20°	64°
25°	65°
30°	66°

As shown by Table 1, the maximum grazing angle increases as the aircraft roll/pitch increases. Figure 8 illustrates the distance past center each of the laser beams reach prior to hitting the ground for grazing angles ranging from 0° to 90°. In Figure 8, a 0-m horizontal distance past center represents the centerline of the image sensor; positive distances are on one side of the image sensor centerline, negative distances are on the other side of the image sensor centerline.

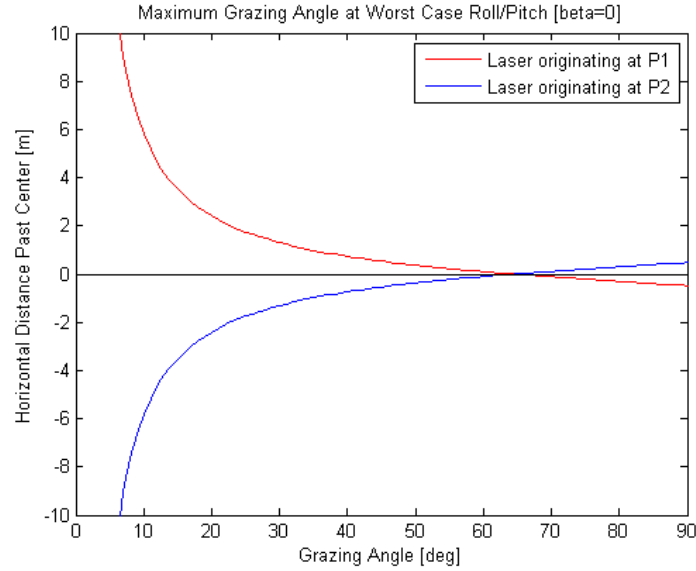


Figure 8 – Maximum grazing angle (1-m altitude, 0° attitude)

### 2.2.2. Ensuring the Laser Beams are within the Field-of-View

The grazing angles must be further constrained such that the point at which the beam of each laser illuminator strikes the landing surface is within the field-of-view (FOV) of the lens used to focus the image sensor. The lens used in this project has a horizontal FOV of 120° and a vertical FOV of 104°. Because the horizontal FOV of the lens is larger than the vertical FOV, this project will align the roll plane of the target aircraft with the horizontal plane of the lens and the pitch plane with the vertical plane of the lens. This alignment gives the system the ability to resolve a larger range of aircraft roll values compared to that of aircraft pitch. It is envisioned that during landing, the target aircraft has the potential to be experiencing significant roll due to strong crosswinds; aircraft pitch, comparatively, should maintain a higher degree of consistency due to the nature of the landing process.

Due to the differences in the horizontal and vertical FOV, different constraints will be developed for each of the two-laser systems comprising the four-laser system. The worst-case altitude with respect to ensuring the beams are within the FOV of the lens is 10 m; as this is the

maximum operating altitude of the system, it presents the largest amount of horizontal distance the beams will travel prior to striking the ground. Table 2 and Table 3 show the lower bounds on the grazing angles,  $\phi_1$  and  $\phi_2$ , for a 10-m altitude above terrain and aircraft attitudes ranging from  $0^\circ$  to  $15^\circ$ .

Table 2 – Minimum grazing angles to ensure beams are within the horizontal FOV

Aircraft Attitude	Grazing Angle
$0^\circ$	$30^\circ$
$5^\circ$	$35^\circ$
$10^\circ$	$40^\circ$
$15^\circ$	$45^\circ$

Table 3 – Minimum grazing angles to ensure beams are within the vertical FOV

Aircraft Attitude	Grazing Angle
$0^\circ$	$37^\circ$
$5^\circ$	$42^\circ$
$10^\circ$	$47^\circ$
$15^\circ$	$52^\circ$

### 2.2.3. Grazing Angle Selection

From the analysis in Sections 2.2.1 and 2.2.2, the grazing angles must be between  $45^\circ$  and  $63^\circ$  for the two-laser system aligned with the horizontal field-of-view of the lens and associated image sensor. Similarly, the two-laser system aligned with the vertical field-of-view must have grazing angles between  $52^\circ$  and  $63^\circ$ . Error analysis was used to determine the grazing angle that provided the best system performance. This error analysis evaluated the absolute maximum altitude and attitude errors as well as the relative maximum altitude error (expressed as a percentage) for  $1^\circ$  increments in the grazing angle for all applicable grazing angles. Table 4 displays the results of this analysis.

Table 4 – Grazing angle error analysis

Grazing Angle [°]	Max Altitude Error [m]	Max Altitude Error [%]	Max Attitude Error [°]
45	0.23	2.34	1
46	0.25	2.45	1
47	0.25	2.46	2
48	0.20	2.01	2
49	0.20	2.07	2
50	0.26	2.56	1
51	0.26	2.62	1
52	0.24	2.39	2
53	0.26	2.55	1
54	0.23	2.27	2
55	0.23	2.27	2
56	0.18	2.06	2
57	0.27	2.69	2
58	0.18	1.97	2
59	0.24	2.35	2
60	0.22	2.24	2
61	0.26	2.63	2
62	0.20	1.98	3
63°	0.21	2.27	9

The maximum and minimum values from Table 4 are presented in Table 5. Presented with these minima and maxima are the corresponding grazing angles, the altitude and attitude estimates generated by the two-laser system, and the actual altitudes and attitudes that the system was estimating.

Table 5 - Maximum and minimum error

	Estimated	Actual	Grazing Angle [°]
Max Altitude Error [m]	10.27	10	57
Min Altitude Error [m]	9.18	9	58
Max Attitude Error [°]	0	9	63
Min Attitude Error [°]	14	15	45

This error analysis shows that the accuracy of the altitude estimates is relatively unaffected by changes in the grazing angle. While this analysis does not provide conclusive evidence at the



effect of the grazing angle on system accuracy, it does indicate when the grazing angles neared the upper limit, the accuracy of the attitude estimates decreased significantly. For this project, grazing angles of  $57^\circ$  were selected. Grazing angles of  $57^\circ$  satisfy the angular constraints presented in Sections 2.2.1 and 2.2.2. Moreover, using angles smaller than the maximum determined in Section 2.2.1 ensures that under normal operation, the beams will never be collocated when they reach the landing surface.

#### **2.2.4. Baseline Lengths**

The baseline lengths between the image sensor and all four laser illuminators will be fixed at 0.5 m for this project. This value was selected to satisfy the constraints of the candidate platform, the 40% YAK-54. Additionally, these baseline lengths allow for adequate system performance while being modest enough to provide reasonable estimates when considering the adaptation of this system to different candidate platforms.

With regard to the baseline lengths, an alternate implementation concept was developed. In this concept, the four-laser system is attached to a flat surface which is then mounted to the underside of the aircraft. To optimize this implementation concept for both the aircraft and the laser altimeter system, a minimization of the baseline lengths was considered. This minimization benefits the aircraft by reducing weight and drag and benefits the laser altimeter by decreasing vibration and the possible flexure of the flat surface to which the laser altimeter is attached.

Using Matlab to simulate system performance, it can be seen from Figure 9 that, at an altitude of 1 m, reducing the baseline lengths to 10 cm does not significantly affect system performance. However, from Figure 10, it can be seen that the same baseline length reduction will produce substantial error in the altitude estimates and erratic estimates of the aircraft attitude. Figure 10 also shows that as the baseline lengths increases, the system accuracy

improves. It should be noted, though, that changes in the baseline lengths affects whether the system satisfies the grazing angle constraints presented in Sections 2.2.1 and 2.2.2.

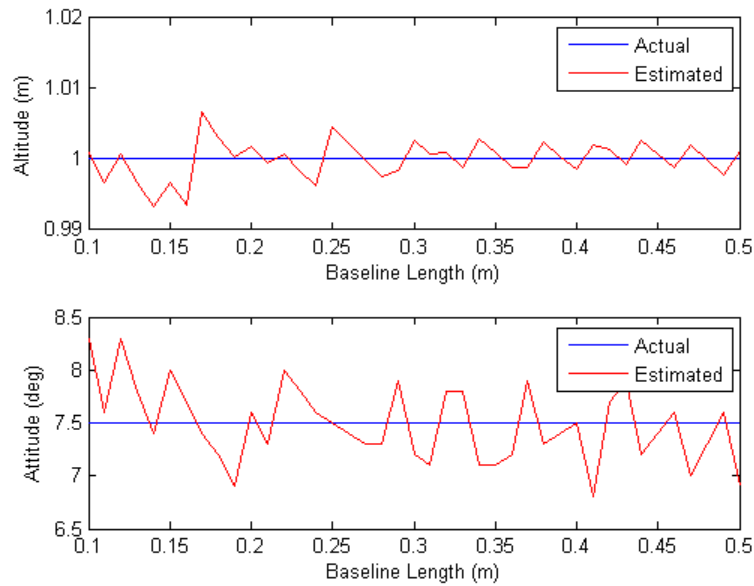


Figure 9 – Baseline length effect on system accuracy (1-m altitude, 7.5° attitude)

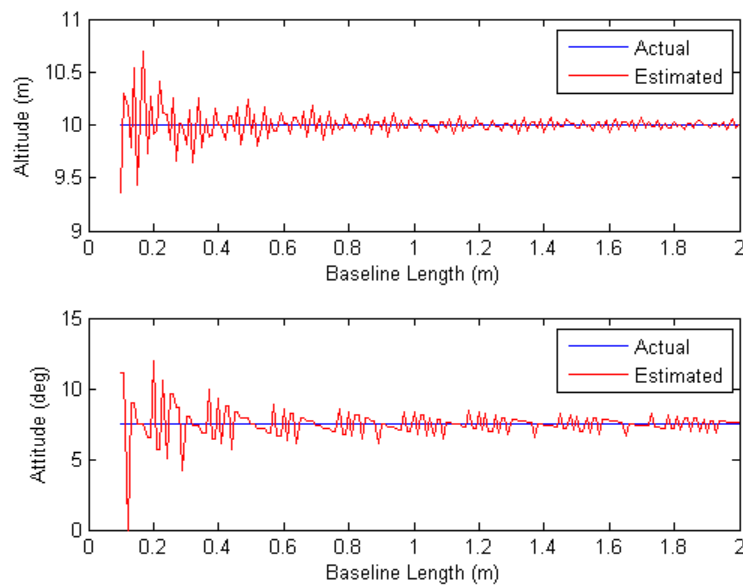


Figure 10 – Baseline length effect on system accuracy (10-m altitude, 7.5° attitude)

### 2.2.5. Wavelength Selection

As a result of their operation in hazardous conditions and environments, the basing options for UAVs are often less than ideal. This means that UAVs can end up landing on concrete, grass, snow, or a number of other surfaces. Thus, for the laser altimeter proposed by this project to perform as expected, the image sensor must be able to accurately detect the points at which the laser beams strike the ground for a variety of landing surfaces. To best detect the laser points on the ground, the wavelength and power of the laser illuminators as well as the spectral characteristics of the image sensor were taken into consideration.

To select the optimum wavelength for the laser illuminators, the quantum efficiency of the image sensor was matched against the reflectivity of a variety of potential landing surfaces. Figure 11 shows the red, green, and blue (RGB) quantum efficiency of the image sensor used in this project [4] and Figure 12 shows the reflectance for a variety of surfaces [7] [10]. From Figure 11 and Figure 12, it was determined that the laser illuminators should have wavelengths of approximately 800 nm.

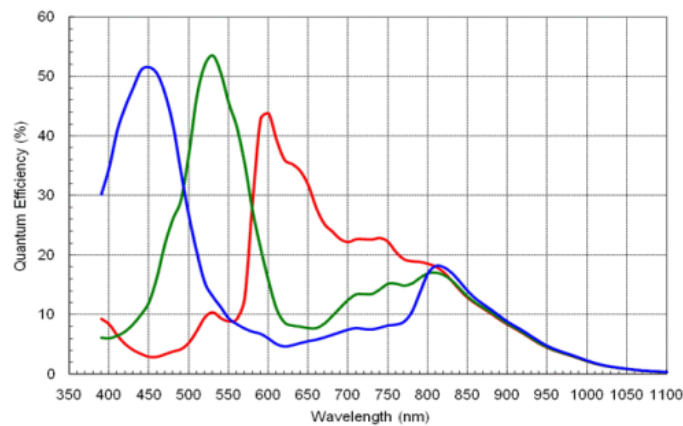


Figure 11 – Image sensor RGB quantum efficiency [4]

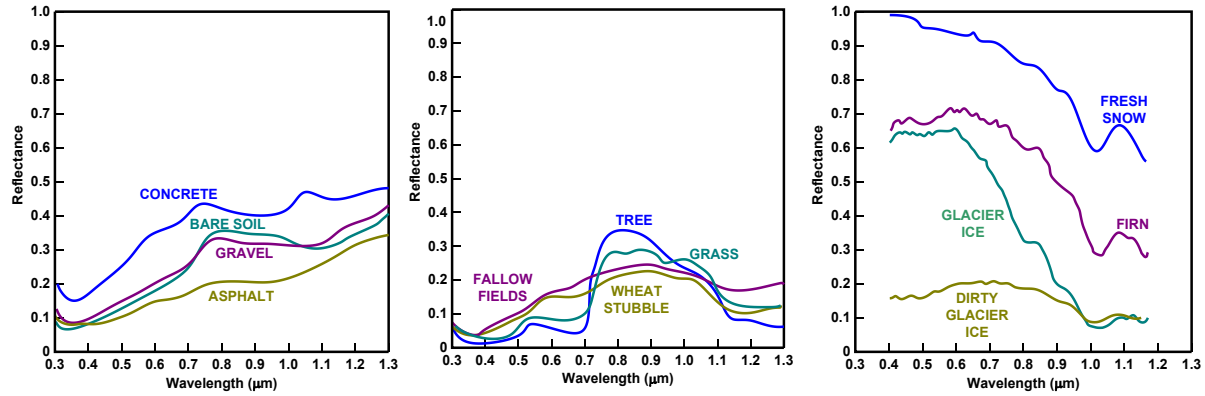


Figure 12 – Inorganic (left), organic (center), and snow/ice (right) surface reflectances [7] [10]

#### 2.2.6. Maximum Laser Power (Eye-Safety)

The output power of the laser illuminators should also be considered. Increasing the power of the laser illuminators increases the signal-to-noise ratio (SNR) of the system, thereby increasing the probability that the most significantly illuminated pixel detected by the image sensor will be a result of the laser illuminators rather than other light sources. However, as this system will be flying overhead when mounted to the underside of a UAV, the beams from the laser illuminators have the potential to strike the eye of an observer. Thus, a maximum laser power was developed such that the system is eye-safe under worst-case operating conditions and reasonable exposure times. Using the equations developed by Wehr and Lohr [16], the output power of each of the laser illuminators used for this project should be limited to 5.23 mW. This maximum was determined by using a laser spot diameter of 9 mm, a range between laser source and observer of 10 m, a laser illuminator wavelength of 780 nm, and exposure time of 0.01 s.

#### 2.2.7. Minimum Laser Power

Similarly, the minimum output power of the laser illuminators should be considered to ensure that the image sensor is able to adequately detect the laser points on the ground. To determine this, the signal-to-noise ratio (SNR) for the image sensor was determined. Using a laser output power of 5 mW, a laser wavelength of 780 nm, a range between the laser source and the ground

of 10 m, a surface reflectivity of 10%, an image sensor quantum efficiency of 20%, and an in-band optical filter attenuation of 3 dB, the SNR for the received scattered light from the laser illuminator at the image sensor is 116.47 dB. This exceeds the  $\text{SNR}_{\text{MAX}}$  value of the image sensor, thus, the laser points on the ground should be effectively detected by the image sensor.

#### **2.2.8. Optical Filter Inclusion**

To ensure the image sensor is able to accurately detect the laser illuminator spots on the ground, a bandpass optical filter was added to the system. This optical filter was placed on the outside of the lens, thereby filtering the light before it reaches the image sensor. The center frequency of the optical filter was matched to the wavelength of the laser illuminators. This ensures that the most significantly illuminated pixel in the image sensor is due to the laser illuminators and not light sources external to the system.

#### **2.2.9. System Simulation**

Prior to system construction, several simulations were created using Matlab to model the performance of the laser altimeter. These simulations verify that the system can achieve the desired accuracy and serve as a means of ensuring the system can be realized on the hardware. Figure 13 and Figure 14 show the outputs of these simulations.

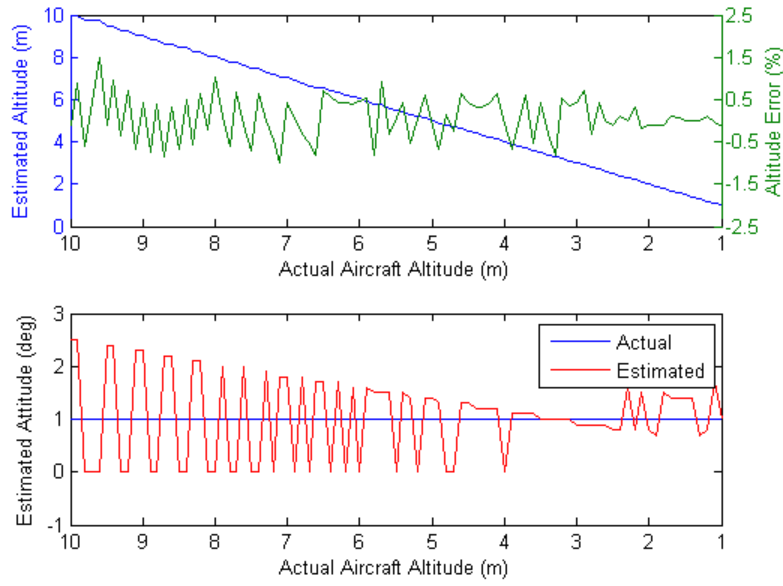


Figure 13 – System simulation at 1° attitude

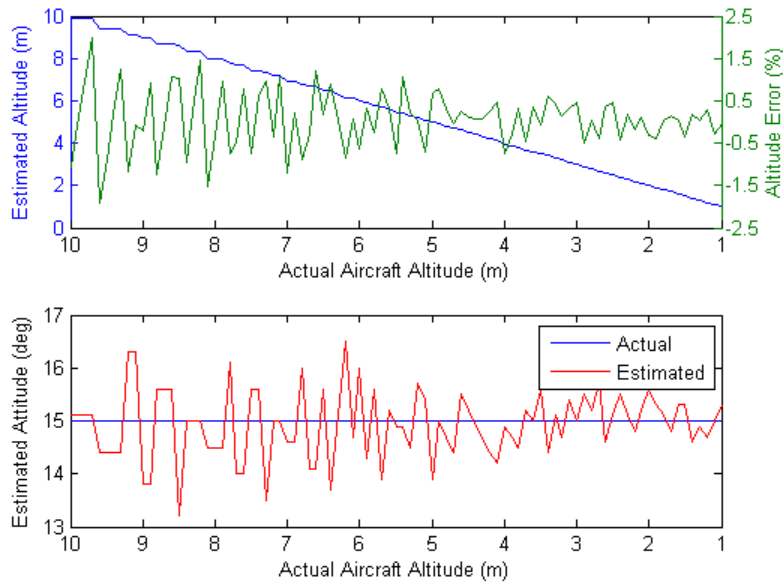


Figure 14 – System simulation at 15° attitude

The error source incorporated into these simulations is the error associated with the rounding of the illuminated pixel in the image sensor to a discrete integer value. As these simulations show, a single pixel error can significantly distort the aircraft attitude. Thus, in order to

accurately resolve aircraft attitude, it is imperative that the correct illuminated pixel be extracted from the image sensor.

The error associated with the rounding of the illuminated pixel in the image sensor to a discrete integer value is a function of the altitude and attitude of the aircraft as well as the physical size of the pixels in the image sensor. This error increases as aircraft altitude increases and also increases as aircraft attitude increases. The maximum error the system will sustain due to this source is then bounded by the operating altitude and attitude of the system. At the maximum operating altitude of 10 m and a maximum aircraft attitude of  $15^\circ$ , the altitude estimate generated by the system has a maximum potential altitude error of 3.33%. Thus, under all operating altitudes and attitudes, the system should satisfy the desired system accuracy presented in Section 1.2

### 3. Presentation of Work

Figure 15 shows a block diagram of the low-altitude laser altimeter developed by this project. The system is comprised of five fundamental components: the processor, image sensor, lens, optical filter and laser illuminators. Each of these components is discussed in more detail in the following sections.

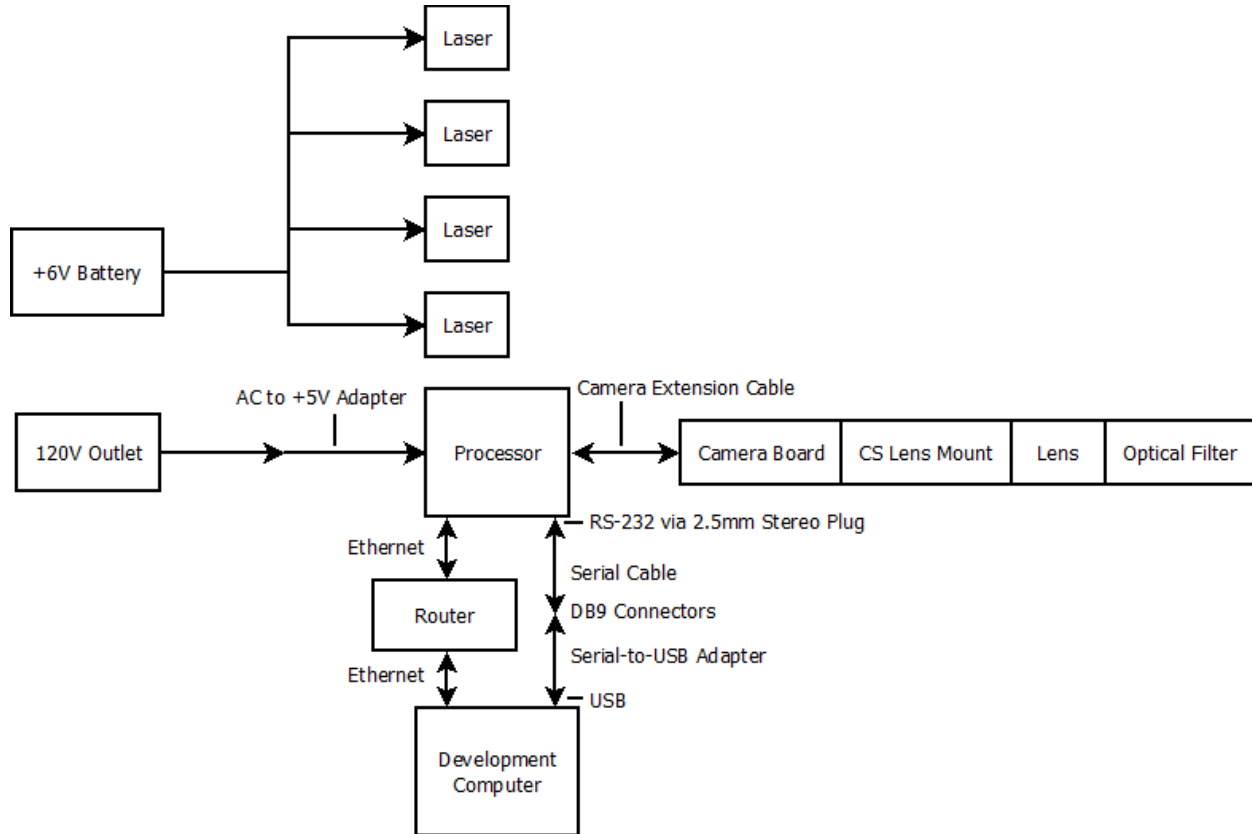


Figure 15 – Laser altimeter system block diagram

#### 3.1. Processor

The primary purpose of the processor used in the laser altimeter is to extract the most significantly illuminated pixels from the image sensor, complete the necessary calculations to generate altitude above terrain, roll, and pitch estimates, and finally output these estimates. To satisfy these requirements, this project used the Leopardboard 365 [5]. The Leopardboard 365 is an off-the-shelf solution produced by Leopard Imaging, Inc. The Leopardboard 365 is comprised



of a TMS320DM365 processor, 256 MB of NAND Flash, 128 MB of DDR2 SDRAM, as well as a number of data interface ports.

A number of considerations were taken into account prior to selecting the Leopardboard 365. These considerations, in order of importance, included interface to image sensor, processing power, software development, operating temperature range, cost, sustainability, and power requirements. One of the primary uses of the Leopardboard 365 is image processing, as Leopard Imaging, Inc. also produces a number of camera boards which are directly compatible with the camera interface on the Leopardboard 365. This direct compatibility simplifies the process of interfacing the processor to the image sensor.

The second consideration taken into account was the processing power of the Leopardboard 365. The TMS320DM365 processor used on the Leopardboard is a 300-MHz ARM9 processor. This processor performs 32-bit and 16-bit instructions and processes 32-bit, 16-bit, and 8-bit data. Moreover, the processor is capable of running a Linux operating system and supports an array of video and audio codecs.

The software development on the Leopardboard 365 was simplified by the availability of a free SDK (software development kit) from RidgeRun. The SDK available from RidgeRun facilitates the setup and installation of a bootloader, Linux kernel, custom C programs, and a variety of associated plug-ins and settings on the Leopardboard 365. Also included in the SDK is a driver for the MT9P031 image sensor used by this project. The SDK was installed on the development computer and communicates with the Leopardboard 365 through serial and Ethernet interfaces.

The Leopardboard 365 has an operating temperature range of -40 °C to +85 °C. This temperature range is robust enough to withstand even some of the more extreme environments

UAVs are capable of operating in. At \$129 the Leopardboard 365 is one of the most inexpensive solutions available. The Leopardboard 365 meets the sustainability requirement as Leopard Imaging, Inc. is actively producing the Leopardboard 365. As much of the hardware and software is compatible, should the Leopardboard 365 reach end-of-life, the upgraded version, the Leopardboard 368, would be a suitable replacement without necessitating a complete redesign of the system. Finally, the Leopardboard 365 consumes a maximum of 2 W when connected to the camera board used by this project.

#### 3.1.1.1. **Software**

Many of the default settings of the SDK provided by RidgeRun were applicable for this project. This includes the driver for the MT9P031, GStreamer plug-ins, and libjpeg plug-ins. Other settings separately enabled through the SDK include floating point emulation and the i2ctools plug-in.

All of the processing for the laser altimeter was completed using a single C program. The program and makefile were created outside of the SDK and then imported into the SDK in order to compile and execute them on the Leopardboard 365. The C program developed for the laser altimeter changes image sensor settings via system calls to the i2ctools plug-in. Color, 720p images are captured from the camera as \*.jpg files using the GStreamer plug-in through the header file “gst/gst.h”. Next, the libjpeg plug-in is accessed through the header file “jpeglib.h” to retrieve the stored image file, decompress it into a grayscale data, and then access the individual pixel information. Finally, the loop for determining aircraft altitude and attitude has a granularity of 1° over the range of -20° to 20°.

### 3.2. Image Sensor

The camera board used for the laser altimeter was the LI-M503 produced by Leopard Imaging, Inc [1]. The LI-M503 costs \$79 and consists of an Aptina MT9P031 CMOS image sensor, supporting circuitry, and CS lens mount [4]. The MT9P031 is a 5-Megapixel (2592 x 1944), 1/2.5", 2D, RGB image sensor. It is capable of capturing full resolution images at a rate of 14 fps or 720p (1280 x 720) images at 60 fps. The LI-M503 plugs directly into the camera interface on the Leopardboard 365.

### 3.3. Lens

Selection of the lens was critical to system performance as the FOV of the lens determines the maximum amount of aircraft attitude the system is able to resolve. However, obtaining a lens with a large FOV requires a shorter EFL (effective focal length), or  $f$  from Equation (1). On the other hand, decreasing the EFL decreases the overall accuracy of the system as the detection of the illuminated pixels exaggerates the angles between the image sensor and the lens.

The lens selected for this project was the SY110M by Theia Technologies [14]. The SY110M is an ultra-wide angle, distortionless, day/night lens that supports multi-megapixel resolutions. The SY110M costs \$245, has an EFL of 1.67 mm, a horizontal FOV of 120°, and a vertical FOV of 104° (FOV values are for 1/2.5" image sensor formats). The lens is available in a CS-mount, so it was directly compatible with the image sensor. Finally, the lens has a depth of field of 176.9 mm to infinity (given a maximum allowable blur size of twice the image sensor pixel pitch) [11].

### 3.4. Optical Filter and Laser Illuminators

The optical filter and laser illuminators selected by this project needed to match each other in wavelength and satisfy the constraints imposed by surface reflectivity and the image sensor

quantum efficiency. Based these constraints in addition to availability and affordability, the Edmund Optics 780-nm optical filter and the US Lasers, Inc. 780-nm laser illuminator were selected for this project [3] [6].

The optical filter from Edmund Optics is a \$225 bandpass interference filter with a 780-nm center wavelength, a 10-nm bandwidth (full width-half max), a blocking wavelength range of 200 nm to 1200 nm, and comes in a 50.8-mm unmounted square. An attempt was made to find a lens with an optical filter mount but no lens/filter/laser combination could be found that satisfied the constraints of the system.

The laser illuminator from US Lasers is a \$56 CW (continuous wave) laser diode module with a 780-nm wavelength, 5-mW output power, and 1.6 mrad of beam divergence. This laser diode module was used for all four laser illuminators on this project. The total power consumption of all four laser diodes is 294 mW.

#### **4. Test and Evaluation**

The test and evaluation portion of this project was extremely important in developing a practical system as two unsuccessful methods were developed for resolving aircraft altitude and attitude prior to arriving at the solution presented by this project. These two unsuccessful methods passed both mathematical inspection and simulation tests, yet were founded upon assumptions that were not valid when applied to the hardware. However, using the test fixture as an additional verification method, the realizable solution presented by this project was found.

##### **4.1. Test Fixture**

The test fixture for this project was constructed from 2-in thick Owens Corning foam insulation. This foam insulation is sold in 4 ft by 8 ft sheets, but, was reduced to 4 ft by 4 ft for this project. The camera board and lens assembly were mounted by drilling a hole in the center of the board. The Leopardboard 365 and camera board were then attached to the top of the test fixture with the CS-mount of the camera board facing downward through the hole in the test fixture. The lens was then attached to the CS-mount from the underside of the test fixture. The optical filter was then suspended over the lens. To install the laser illuminators, holes slightly smaller than the diameter of laser illuminators were drilled in the test fixture. Then, the laser illuminators were pressed into the test fixture from the underside. After adjustments to the grazing angles were made, the laser illuminators were set in place using hot glue. Figure 16 through Figure 21 show various aspects of the test fixture. Figure 22 shows the laser illuminator spots on the ground (as captured by an observer standing to the side of the test fixture).



Figure 16 – Test fixture and hardware

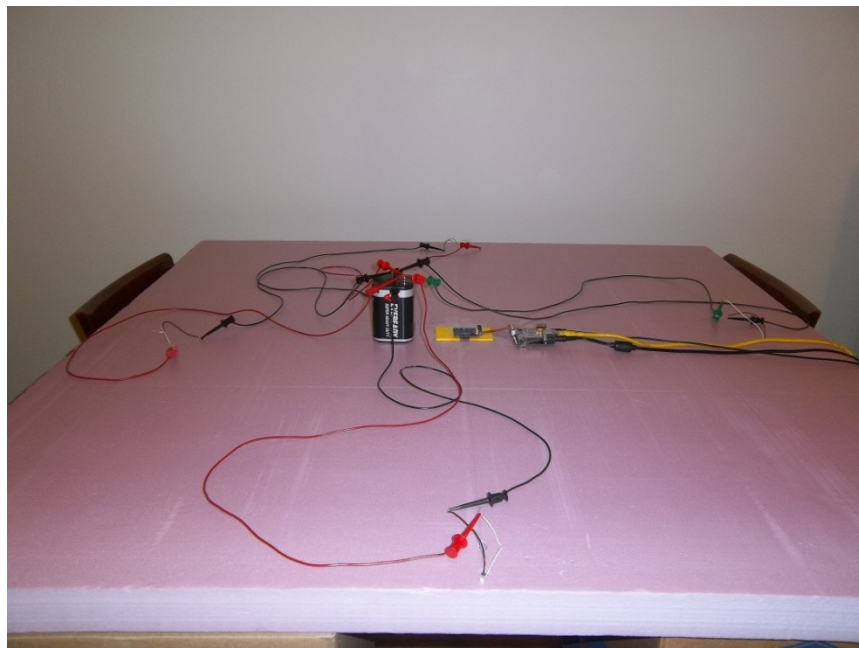


Figure 17 – Hardware configuration on the test fixture

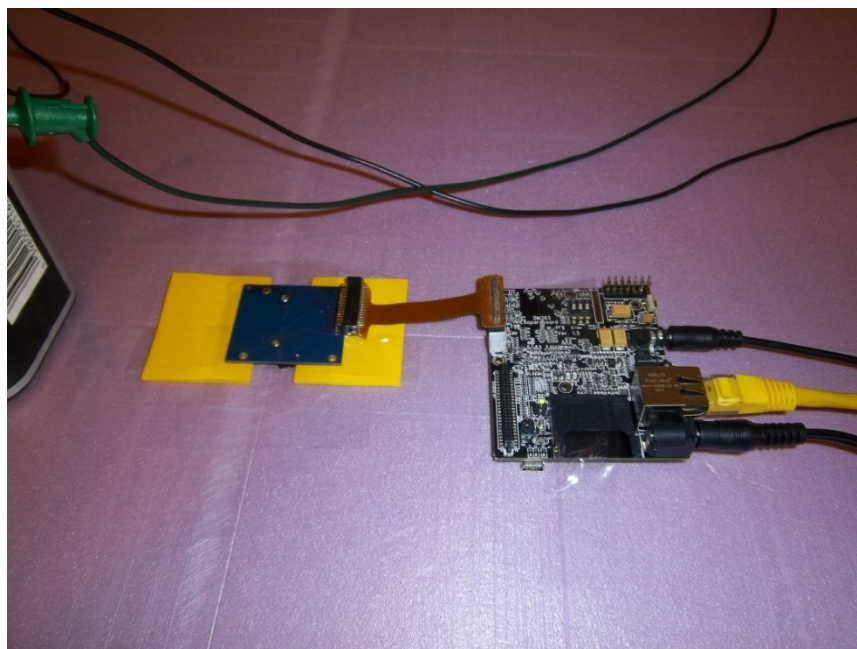


Figure 18 – Leopardboard 365 on the test fixture



Figure 19 – Lens on the underside of the test fixture





Figure 20 – Optical filter covering the lens



Figure 21 – Laser illuminators embedded in the test fixture





Figure 22 – Laser illuminators on the ground as captured by an observer standing to the side of the test fixture

An attempt was made to construct the test fixture according to the specifications developed by this project; however, due to the material used in the test fixture and the nature of the construction process itself, many of the specifications were not perfectly attained. Table 6 outlines the final specifications of the test fixture. Additionally, the camera extension cable prevented the mounting of the camera board flush with the test fixture; one side of the camera board is approximately 3 mm further away from the test fixture than the other. Thus, the center of the image sensor is not perfectly aligned with the surface normal.

Table 6 – Final specifications of the test fixture

Specification	Value
Roll Plane Baseline Lengths	55 cm
Pitch Plane Baseline Lengths	48 cm
Roll Plane $\varphi_1$	58°
Roll Plane $\varphi_2$	58°
Pitch Plane $\varphi_1$	55°
Pitch Plane $\varphi_2$	55°

#### 4.2. Processor Speed

To develop a baseline execution speed, the ability of the processor to handle a variety of mathematical operations was tested. As Table 7 shows, the Leopardboard 365 is capable of more than 100,000 complex arithmetic operations per second. Next, the execution speeds for the operations necessary to this project were tested. Table 8 shows the results of this testing. The operation “GStreamer Setup” is not included in the total execution time at the bottom of Table 8 as it must only be executed once. “GStreamer Setup” involves allocating the GStreamer pipeline and setting up the associated parameters. This operation must be executed prior to the capture of the first image from the image sensor. The remainder of the operations in Table 8 must be executed each time an altitude and attitude estimate is generated by the system.

Table 7 – Processor operation execution times

Operation	Execution Time [ $\mu$ s]
Sine	9
Cosine	7
Tangent	5
Arcsine	5
Arccosine	5
Arctangent	5
Multiplication	5
Division	4

Table 8 – System execution time

Operation	Execution Time [ms]
GStreamer Setup	1307.701
Image Capture	764.266
Image Decompression	3.373
Image Search	237.617
Image Destroy	0.293
Altitude and Attitude Calculations	15.666
Total	1021.215

Table 8 shows that the majority of the system execution time is consumed by the “Image Capture” operation. This operation involves setting the GStreamer pipeline to “playing”, retrieving the data from the image sensor, saving the data as a \*.jpg file, and then stopping the GStreamer pipeline. The second longest operation is the “Image Search” operation. This operation involves reading the pixel data from the saved \*.jpg file and determining the coordinates of the most significantly illuminated pixel. The total system execution time of 1021.215 ms corresponds to an output data rate of 0.98 Hz. This value does not meet the output data rate specification developed in Section 1.2.

#### 4.3. Accuracy

Two test scenarios were conducted to evaluate the performance of the system. Test scenario #1 was conducted at a zero attitude altitude of 1.029 m. Test scenario #2 was conducted at a zero

attitude altitude of 1.670 m. Both scenarios were conducted indoors, in fluorescent lighting, over a carpeted surface. Table 9 and Table 10 show the results of these tests. The optical filter was removed during test scenario #2. This was due to the fact that with the optical filter in place, the laser points on the ground were not bright enough for the image sensor to detect. Figure 23 shows the laser illuminator spots on the ground (as captured by the image sensor during test scenario #2).

Table 9 – System outputs during test scenario #1

Actual Roll [°]	Est. Roll [°]	Actual Pitch [°]	Est. Pitch [°]	Actual Alt. [m]	Est. Alt. [m]
12.48	12	0	0	1.161	1.19
9.59	10	0	-1	1.131	1.15
6.88	6	0	-1	1.102	1.11
0	0	0	0	1.029	1.05
-6.88	-9	0	2	1.102	1.12
-9.59	-12	0	-1	1.131	1.17
-12.48	-16	0	-3	1.161	1.21
0	0	12.48	16	1.161	1.20
0	0	9.59	11	1.131	1.16
0	-1	6.88	7	1.102	1.13
0	0	-6.88	-5	1.102	1.14
0	-1	-9.59	-9	1.131	1.17
0	0	-12.48	-12	1.161	1.22
-3.43	-4	-3.43	-5	1.062	1.11
-4.78	-4	-4.78	-7	1.080	1.15

Table 10 – System outputs during test scenario #2

Actual Roll [°]	Est. Roll [°]	Actual Pitch [°]	Est. Pitch [°]	Actual Alt. [m]	Est. Alt. [m]
0	2	0	1	1.670	1.68
-6.88	-5	0	-1	1.743	1.74
-9.59	-12	0	0	1.772	1.78
-12.48	-14	0	-2	1.802	1.84
0	0	6.88	7	1.743	1.74
0	1	9.59	10	1.772	1.78
0	0	12.48	12	1.802	1.81



Figure 23 - Laser illuminators on the ground as captured by the image sensor

Table 11 shows the average error of the altitude and attitude estimates generated by the system during test scenarios #1 and #2. These average error values meet the accuracy specifications developed in Section 1.2.

Table 11 – Average error of system outputs during test scenarios #1 and #2

System Output	Average Error
Altitude	2.28%
Roll	1.00°
Pitch	1.13°

## 5. Conclusion

This project developed a low-altitude laser altimeter system to assist UAV autolandings. With an altitude accuracy of 2.28% and an attitude accuracy of approximately  $1^\circ$ , the system was able to meet the altitude accuracy requirement of 5% and the attitude accuracy requirement of  $3^\circ$ . If processing time was not a factor, the system could provide more precise estimates by increasing the granularity of the  $\beta$  search solution approach (e.g. from  $1^\circ$  to  $0.1^\circ$  increments). However, the system was unable to attain the desired output data rate of 10 Hz. The output data rate of the system was approximately 1 Hz, which indicates that improvements must be made before the system is ready for deployment. Additionally, the optical filter had to be removed during test scenario #2 in order for the image sensor to detect the laser illumination points on the ground. Because the SNR for system is ample, this implies that the optical interference filter could be having an adverse effect on the performance of the system due to the off-normal input angles.

## 6. Future Work

There are a number of other action items in regards to the continuance of this project.

Foremost, to satisfy output data rate requirement of 10 Hz, an alternate method of obtaining the pixel information from the camera should be investigated. As previously stated, the image sensor is capable of 14 fps at full resolution, so, the limiting factor is not the speed of the image sensor, but rather the overhead encountered when using the GStreamer plug-in to obtain the data from the image sensor. Two methods could improve this rate; the first is to access the raw data output from the GStreamer plug-in rather than using GStreamer to save the image sensor data as a \*.jpg file. The second method is to create a driver that can access the pixel data directly and bypass the use of the GStreamer and libjpeg plug-ins altogether. Alternatively, a monochrome image sensor could replace the color image sensor used in this project. This would reduce the amount of data to be extracted from the image sensor, which could, in turn, increase the output data rate. However, system accuracy could be affected if the pixel size is increased.

In a corollary to the first improvement, the second improvement that could be made in this project is to increase the output resolution of the image sensor. The accuracy of the attitude estimate is particularly sensitive to the relative pixel size. Increasing the output resolution of the camera decreases the relative pixel size thereby increasing the accuracy of the attitude estimates. The image sensor is capable of 2592 x 1944 images; however, as far as it is currently known, the libjpeg plug-in requires each line of the image to be read consecutively. If specific rows/columns of the image could be read instead, the resolution of the retrieved image could be improved and the overall output data rate of the system could be improved.

Once the software is finalized and the output data rate has been improved, the next step would be to deploy the Leopardboard 365 remotely. This would include the addition of an SD

card to store the file system, an output data log to record the altitude and attitude estimates, and potentially an on-board or remote display to assist the testing process. The final step in the deployment would be to interface the output of the system to the existing autopilot.

As the system was unable to detect the illuminated spots on the ground when the altitude was increased to 1.670 m, either laser illuminators with a larger output power or a different optical filter should be selected. As the interference optical filter used by this system is intended for collimated input, selecting a colored glass optical filter might increase the performance of the system by allowing more of the off-normal scattered light from the laser illuminator to reach the image sensor. Selecting laser illuminators with a larger output power is also a feasible solution. But, since the power of the laser illuminators used by this project is close to the maximum eye-safe power, additional precautions should be taken to protect observers.

Finally, one of the greatest improvements that could be made in the accuracy of this project would be to construct a more precise and robust test fixture. Most specifically, a mount needs to be created for the laser illuminators. Obtaining accurate grazing angles and the aligning the image sensor was an incredibly tedious and inaccurate task. To make the test fixture more robust, it needs the ability to test the system at altitudes up to 10 m and with attitude increments of  $1^\circ$ . Additionally, the test system should be tested on different surfaces (concrete, grass, etc.) to verify the comprehensive operation of the system.



## 7. References

- [1] *5M Camera Board LI-5M03*. Leopard Imaging, Inc. Datasheet.
- [2] *hc-profi\_manual\_v2.5.pdf*. HeliCommand, June 2010. Web.  
<<http://www.helicommand.com>>.
- [3] "Infrared Bandpass Interference Filters." *edmundoptics.com*. Edmund Optics Worldwide.  
Web. <<http://www.edmundoptics.com/optics/optical-filters/bandpass-filters/infrared-bandpass-interference-filters/3430>>.
- [4] *MT9P031: 1/2.5-Inch 5Mp CMOS Digital Image Sensor*. Aptina. 2005. Datasheet.
- [5] *Leopardboard 365 Hardware User Guide*. Leopard Imaging, Inc. 2010. Datasheet.
- [6] *M780-5: 780nm 5mW Laser Module*. US Lasers, Inc. Datasheet.
- [7] Maurer, John. *Retrieval of Surface Albedo from Space*. Tech. University of Hawaii at Manoa, Dec. 2002. Web. <<http://www2.hawaii.edu/~jmaurer/albedo/>>.
- [8] *MRA-Type-2.pdf*. Roke Manor Research Limited, Aug. 2011. Web.  
<<http://www.roke.co.uk>>.
- [9] *MT9P031: 1/2.5-Inch 5Mp CMOS Digital Image Sensor*. Aptina. 2005. Datasheet.
- [10] NASA. *The Landsat Tutorial Workbook: Basics of Satellite Remote Sensing*. Washington DC: Scientific and Technical Information Branch, 1982. Web.  
<<http://landsat.gsfc.nasa.gov/education/tutorials.html>>.
- [11] *Optical Wizards*. Sunex, 2012. Web. <<http://www.optics-online.com/wizards.asp>>.
- [12] Reich, Stefan. Optical Sensing System and System for Stabilizing Machine-Controllable Vehicles. Patent 7400950. 15 July 2008.
- [13] *srf08.pdf*. Devantech, Feb. 2003. Web. <<http://www.jk-sensor.com>>.

- [14] *SY110 Megapixel Lens: Ultra-wide – Day/Night – No Distortion*. Theia Technologies. Datasheet.
- [15] "Universal Laser Sensor (ULS) Specifications." *Lasertetch.com*. Laser Technology, Inc. Web. <<http://www.lasertech.com/Universal-Laser-Sensor-Specs.aspx>>.
- [16] Wehr, Aloysius, and Uwe Lohr. "Airborne Laser Scanning - an Introduction and Overview." *ISPRS Journal of Photogrammetry & Remote Sensing* (1999): 68-82. Print.

## 8. Appendices

### A. Mathematical Analysis

#### A.1. Optical Triangulation

See Figure 1 for variable definitions.

$$\tan \theta = \frac{u}{f} \quad (9)$$

$$\tan \theta = \frac{x}{D} \quad (10)$$

$$\frac{u}{f} = \frac{x}{D} \quad (11)$$

$$xf = uD \quad (12)$$

#### A.2. One-Laser Altitude Above Terrain

See Figure 2 for variable definitions.

$$h = \frac{B}{\tan \theta_1 - \tan \theta_0} \quad (13)$$

$$h = \frac{B}{\frac{B + x_1}{h} - \frac{x_1}{h}} \quad (14)$$

$$h = \frac{B}{\frac{B}{h}} \quad (15)$$

#### A.3. Laser Offset from the Horizontal Plane of the Camera

See Figure 3 for variable definitions.

$$\cot \varphi_1 = \frac{b}{a} \quad (16)$$

$$b = a \cot \varphi_1 \quad (17)$$

$$B' = B - b = B - a \cot \varphi_1 \quad (18)$$

#### A.4. Two- Laser Altitude Above Terrain Derivation

See Figure 4 for variable definitions.

$$\theta_1 = 90^\circ - \varphi_1 + \beta \quad (19)$$

$$\theta_2 = 90^\circ - \varphi_2 - \beta \quad (20)$$

$$\varphi_a = 90^\circ - \theta_a + \beta \quad (21)$$

$$\varphi_b = 90^\circ - \theta_b - \beta \quad (22)$$

$$\theta'_a = \theta_a - \beta \quad (23)$$

$$\theta'_b = \theta_b + \beta \quad (24)$$

$$x_1 = -B_1 \cos \beta + (h - B_1 \sin \beta) \tan \theta_1 = -B_1 \cos \beta + (h - B_1 \sin \beta) \cot(\varphi_1 - \beta) \quad (25)$$

$$-x_2 = -B_2 \cos \beta + (h + B_2 \sin \beta) \tan \theta_2 = -B_2 \cos \beta + (h + B_2 \sin \beta) \cot(\varphi_2 + \beta) \quad (26)$$

$$\cot(\varphi_1 - \beta) = \frac{x_1 + B_1 \cos \beta}{h - B_1 \sin \beta} = \frac{x_1}{h - B_1 \sin \beta} + \frac{B_1 \cos \beta}{h - B_1 \sin \beta} \quad (27)$$

$$\cot(\varphi_2 + \beta) = \frac{-x_2 + B_2 \cos \beta}{h + B_2 \sin \beta} = \frac{-x_2}{h + B_2 \sin \beta} + \frac{B_2 \cos \beta}{h + B_2 \sin \beta} \quad (28)$$

$$\tan \theta_a = \frac{x_1}{h} = \left(1 - \frac{B_1 \sin \beta}{h}\right) \cot(\varphi_1 - \beta) - \frac{B_1}{h} \cos \beta \quad (29)$$

$$\frac{x_1}{h} = \left(1 - \frac{B_1 \sin \beta}{h}\right) \left(\frac{\cot \varphi_1 \cot \beta + 1}{\cot \beta - \cot \varphi_1}\right) - \frac{B_1}{h} \cos \beta \quad (30)$$

$$\frac{x_1}{h} = \cot(\varphi_a - \beta) = -\frac{B_1}{h} \cos \beta + \left(1 - \frac{B_1 \sin \beta}{h}\right) \cot(\varphi_1 - \beta) \quad (31)$$

$$\tan \theta_b = \frac{-x_2}{h} = \left(1 + \frac{B_2 \sin \beta}{h}\right) \cot(\varphi_2 + \beta) - \frac{B_2}{h} \cos \beta \quad (32)$$

$$\frac{-x_2}{h} = \left(1 + \frac{B_2 \sin \beta}{h}\right) \left(\frac{\cot \varphi_2 \cot \beta - 1}{\cot \beta + \cot \varphi_2}\right) - \frac{B_2}{h} \cos \beta \quad (33)$$

$$\frac{-x_2}{h} = \cot(\varphi_b + \beta) = -\frac{B_2}{h} \cos \beta + \left(1 + \frac{B_2 \sin \beta}{h}\right) \cot(\varphi_2 + \beta) \quad (34)$$

$$h_1 = \frac{B_1 \sin \beta \cot(\beta - \varphi_1) - B_1 \cos \beta}{\cot(\beta - \varphi_1) - \cot(\beta - \varphi_a)} \quad (35)$$

$$h_2 = \frac{B_2 \sin \beta \cot(\beta + \varphi_2) - B_2 \cos \beta}{\cot(\beta + \varphi_b) - \cot(\beta + \varphi_2)} \quad (36)$$

$$h = \frac{h_1 + h_2}{2} \quad (37)$$

#### A.5. Maximum Laser Power (Eye-Safety)

$$A_I = \pi * R^2 \quad (38)$$

$$A_I = \pi * (4.5 * 10^{-3})^2 \quad (39)$$

$$A_I = 6.36 * 10^{-5} \quad (40)$$

Where  $R$  is the radius of the laser spot on the ground at an altitude of 10 m [m] and  $A_I$  is the area the laser spot on the ground at an altitude of 10 m [m<sup>2</sup>].

$$P_{av} = 18 * 10^{\frac{\lambda-700}{500}} * t^{-0.25} * A_I \quad (41)$$

$$P_{av} = 18 * 10^{\frac{780-700}{500}} * (0.01)^{-0.25} * (6.36 * 10^{-5}) \quad (42)$$

$$P_{av} = 5.23 \quad (43)$$

Where  $\lambda$  is the wavelength of the laser [nm],  $t$  is the exposure time [s], and  $P_{av}$  is the average continuous-wave (CW) output power of the laser [mW].

#### A.6. Minimum Laser Power

$$P_r = \frac{P_t}{2\pi * R^2} A_{eff} \rho \quad (44)$$

$$P_r = \frac{5 * 10^{-3}}{2\pi * 10^2} (1.453 * 10^{-4}) * 0.1 \quad (45)$$

$$P_r = 1.156 * 10^{-10} \quad (46)$$

Where  $P_t$  is the transmitted power of the laser [W],  $R$  is the range between the laser source and the ground [m],  $A_{eff}$  is the area of the lens [m<sup>2</sup>],  $\rho$  is the surface reflectance, and  $P_r$  is the power received at the image sensor [W].

$$I_r = \frac{P_r * \lambda}{h * c} Q \quad (47)$$

$$I_r = \frac{(1.156 * 10^{-10}) * (780 * 10^{-9})}{(6.626 * 10^{-34}) * (3 * 10^8)} (0.2) \quad (48)$$

$$I_r = 9.406 * 10^7 \quad (49)$$

Where  $P_r$  is the power received at the image sensor [W],  $\lambda$  is the wavelength of the laser [m],  $h$  is Planck's constant [J·s],  $c$  is the speed of light [m/s],  $Q$  is the quantum efficiency of the image sensor at 780 nm, and  $I_r$  is the received current in the image sensor [electrons/s].

$$SNR = 20 \log \left( \frac{I_r}{I_n} \right) - F_{att} \quad (50)$$

$$SNR = 20 \log \left( \frac{9.406 * 10^7}{100} \right) - 3 \quad (51)$$

$$SNR = 116.47 \quad (52)$$

Where  $I_r$  is the received current in the image sensor [electrons/s],  $I_n$  is the dark current noise of the image sensor [electrons/s],  $F_{att}$  is the in-band attenuation of the optical filter [dB], and SNR is the signal-to-noise ratio at the image sensor [dB].

## B. System Code (C)

```
/*-----
```

File: main.c

Description:

- Tests the speed of processor operations.
- Sets up camera parameters.
- Captures images using GStreamer.
- Processes images to determine altitude, roll, and pitch.

Author: Nic Bergmann

```
-----*/
```

```
/*-----
```

INCLUDES

```
-----*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/time.h>
#include <jpeglib.h>
#include <gst/gst.h>
```

```
/*-----
```

VARIABLES

```
-----*/
```

```
static int capture_delay = 0; //specifies the number of times to loop
                               //through the timeout function
```

```
/*-----
```

PROCEDURES

```
-----*/
```

```
gboolean timeout (gpointer data) //timeout to ensure GStreamer returns
{
    //from image capture process
    capture_delay++;
    g_print("Timeout Called %d Times\n",capture_delay);
```

```
if (capture_delay == 3)
{
    capture_delay = 0;
    g_main_loop_quit((GMainLoop*) data);
    return FALSE;
}
```

```
return TRUE;
```

```

}

double cot (double arg) //cotangent function
{
return 1/tan(arg);
}

/*-----
MAIN PROGRAM
-----*/
int main (int argc, char* argv[])
{
//Timing Variables
struct timeval start,end;
long          seconds,useconds;

//Math Functions
gettimeofday(&start,NULL);

sin(1.57);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for sine: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

cos(1.57);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for cosine: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

tan(1.57);

gettimeofday(&end,NULL);

```



```

seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for tangent: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

asin(.785);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for arcsine: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

acos(.785);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for arccosine: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

atan(.785);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for arctan: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

1.57*.785;

```

```

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for multiplication: %ld seconds, %ld microseconds\n",
seconds,
useconds);

```

```

gettimeofday(&start,NULL);

```

```

1.57/.785;

```

```

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for division: %ld seconds, %ld microseconds\n",
seconds,
useconds);

```

```

//Setup Camera Parameters

```

```

system("i2cset -f -y 1 0x48 0x020 0x4040 w");

```

```

//GStreamer

```

```

gettimeofday(&start,NULL);

```

```

GMainLoop *loop;
GstElement *pipeline,*cam_src,*flt,*encoding,*sink;

```

```

//Initialize

```

```

gst_init(&argc,&argv);
loop = g_main_loop_new(NULL,FALSE);

```

```

//Create GStreamer elements

```

```

pipeline = gst_pipeline_new("camera");
cam_src = gst_element_factory_make("v4l2src","cam_src");
flt = gst_element_factory_make("capsfilter","flt");
encoding = gst_element_factory_make("dmaienc_jpeg","encoding");
sink = gst_element_factory_make("filesink","sink");

```

```

if (!pipeline || !cam_src || !encoding || !sink)
{
g_printerr("One element could not be created. Exiting.\n");
return -1;
}

```

```

//Setup the pipeline
g_object_set(G_OBJECT (cam_src),
"always-copy",FALSE,
"num-buffers",1,
"chain-ipipe",TRUE,
NULL);

g_object_set(G_OBJECT (flt),"caps",
gst_caps_new_simple("video/x-raw-yuv",
"format",GST_TYPE_FOURCC,GST_MAKE_FOURCC('U','Y','V','Y'),
"width",G_TYPE_INT,1280,
"height",G_TYPE_INT,720,
NULL),
NULL);

g_object_set(G_OBJECT (sink),"location","image.jpg",NULL);

gst_bin_add_many(GST_BIN (pipeline),cam_src,flt,encoding,sink,NULL);
gst_element_link_many(cam_src,flt,encoding,sink,NULL);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for GStreamer setup: %ld seconds, %ld microseconds\n",
seconds,
useconds);

//Image Search Setup
unsigned int r;
int width, height;

struct jpeg_decompress_struct cinfo;
struct jpeg_error_mgr jerr;

FILE* infile; //source file
JSAMPARRAY pJpegBuffer; //output row buffer
int row_stride; //physical row width in output buffer

int x;
int y = 0;

unsigned int x_min = 620;
unsigned int x_max = 660;

```

```

unsigned int x_mid = 640;
unsigned int y_mid = 440;

unsigned int x_left = 0;
unsigned int x_right = 0;
unsigned int x_bottom = 0;
unsigned int x_top = 0;

unsigned int y_left = 0;
unsigned int y_right = 0;
unsigned int y_bottom = 0;
unsigned int y_top = 0;

unsigned int r_left = 0;
unsigned int r_right = 0;
unsigned int r_bottom = 0;
unsigned int r_top = 0;

float h_roll = 1;
float h_pitch = 1;
float h1,h2,h;
float roll = 0;
float pitch = 0;
float theta_a,theta_b;
float phi_a,phi_b;
float u1,u2;

int index;
float beta;
float loop_start = -20;
float inc = 1;
float loop_end = 20;
float length = (loop_end-loop_start)/inc;

float diff;
float diff_stored = 100;

float pi = 3.14159;
float K = pi/180;
float EFL = 1.68;

float horz_B1 = 0.55;
float horz_B2 = 0.55;
float vert_B1 = 0.48;
float vert_B2 = 0.48;

```

```

float horz_phi_1 = 58;
float horz_phi_2 = 58;
float vert_phi_1 = 55;
float vert_phi_2 = 55;

float horz_imager_size = 5.70;
float vert_imager_size = 4.28;

unsigned int horz_pixels = 1280;
unsigned int vert_pixels = 720;

float horz_eff_pixel = horz_imager_size/horz_pixels;
float vert_eff_pixel = vert_imager_size/vert_pixels;

float horz_f = EFL/horz_eff_pixel;
float vert_f = EFL/vert_eff_pixel;

//Loop Through Capture & Search
int capture_loop;

for (capture_loop = 0; capture_loop < 1; capture_loop++)
{
//Capture
gettimeofday(&start,NULL);

//Set the pipeline to "playing" state
g_print("Setting pipeline to playing\n");
gst_element_set_state (pipeline,GST_STATE_PLAYING);

//Capture
g_print("Running...\n");
g_timeout_add(100,timeout,loop);
g_main_loop_run(loop);

//Out of the main loop, clean up
g_print ("Returned, stopping playback\n");
gst_element_set_state (pipeline,GST_STATE_NULL);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for image capture: %ld seconds, %ld microseconds\n",
seconds,
useconds);

```

```

//Search
if ((infile = fopen("image.jpg","rb")) == NULL)
{
    fprintf(stderr, "Can't open %s\n", "image.jpg");
    return 0;
}

gettimeofday(&start,NULL);

cinfo.err = jpeg_std_error(&jerr);
jpeg_create_decompress(&cinfo);
jpeg_stdio_src(&cinfo, infile);
(void) jpeg_read_header(&cinfo, TRUE);

cinfo.out_color_space = JCS_GRAYSCALE; //sets output_components = 1
(void) jpeg_start_decompress(&cinfo);    //default the image is RGB with
width = cinfo.output_width;             //output_components = 3
height = cinfo.output_height;

printf("Image Width: %i\n",width);
printf("Image Height: %i\n",height);

row_stride = width * cinfo.output_components;
printf("Row Stride: %i\n",row_stride);

pJpegBuffer = (*cinfo.mem->alloc_sarray)
((j_common_ptr) &cinfo, JPOOL_IMAGE, row_stride, 1);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for image decompress: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

while (cinfo.output_scanline < height)
{
    y++;

    (void) jpeg_read_scanlines(&cinfo,pJpegBuffer,1);

    for (x = 0; x < width-1; x++)
    {

```

```

r = pJpegBuffer[0][cinfo.output_components*x];

if ((x > x_min) && (x < x_max))
{
if (y < y_mid)
{
if (r > r_top)
{
r_top = r;
x_top = x;
y_top = y-1;
}
}
else
{
if (r >= r_bottom)
{
r_bottom = r;
x_bottom = x;
y_bottom = y-1;
}
}
else if (x < x_mid)
{
if (r > r_left)
{
r_left = r;
x_left = x;
y_left = y-1;
}
}
else
{
if (r >= r_right)
{
r_right = r;
x_right = x;
y_right = y-1;
}
}
}

printf("\n");
printf("Left Red: %i\n",r_left);

```

```

printf("Left X: %i\n",x_left);
printf("Left Y: %i\n",y_left);

printf("Right Red: %i\n",r_right);
printf("Right X: %i\n",x_right);
printf("Right Y: %i\n",y_right);

printf("Bottom Red: %i\n",r_bottom);
printf("Bottom X: %i\n",x_bottom);
printf("Bottom Y: %i\n",y_bottom);

printf("Top Red: %i\n",r_top);
printf("Top X: %i\n",x_top);
printf("Top Y: %i\n",y_top);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for image search: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

fclose(infile);
(void) jpeg_finish_decompress(&cinfo);
jpeg_destroy_decompress(&cinfo);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;
printf(
"Elapsed time for image destroy: %ld seconds, %ld microseconds\n",
seconds,
useconds);

gettimeofday(&start,NULL);

//Roll
printf("\n");
printf("Roll\n");

u1 = abs(x_mid - x_right);
u2 = abs(x_mid - x_left);

```



```

printf("u1: %f\n",u1);
printf("u2: %f\n",u2);

theta_a = (atan(u1/horz_f))/K;
theta_b = (atan(u2/horz_f))/K;

printf("theta_a: %f\n",theta_a);
printf("theta_b: %f\n",theta_b);

phi_a = 90 - theta_a;
phi_b = 90 - theta_b;

printf("phi_a: %f\n",phi_a);
printf("phi_b: %f\n",phi_b);

for (index = 0; index <= length; index++)
{
    beta = loop_start + index*inc;

    h1 = (horz_B1*sin(beta*K)*cot((beta-horz_phi_1)*K)-horz_B1*cos(beta*K))/
        (cot((beta-horz_phi_1)*K)-cot((beta-phi_a)*K));
    h2 = (horz_B2*sin(beta*K)*cot((beta+horz_phi_2)*K)-horz_B2*cos(beta*K))/
        (cot((beta+phi_b)*K)-cot((beta+horz_phi_2)*K));

    diff = fabs(h1-h2);

    if (diff < diff_stored)
    {
        diff_stored = diff;
        h_roll = (h1 + h2)/2;
        roll = beta;
    }
}

//Pitch
printf("\n");
printf("Pitch\n");

u1 = abs(y_mid - y_top);
u2 = abs(y_mid - y_bottom);

printf("u1: %f\n",u1);
printf("u2: %f\n",u2);

theta_a = (atan(u1/vert_f))/K;
theta_b = (atan(u2/vert_f))/K;

```

```

printf("theta_a: %f\n",theta_a);
printf("theta_b: %f\n",theta_b);

phi_a = 90 - theta_a;
phi_b = 90 - theta_b;

printf("phi_a: %f\n",phi_a);
printf("phi_b: %f\n",phi_b);

diff_stored = 100;

for (index = 0; index <= length; index++)
{
    beta = loop_start + index*inc;

    h1 = (vert_B1*sin(beta*K)*cot((beta-vert_phi_1)*K)-vert_B1*cos(beta*K))/
        (cot((beta-vert_phi_1)*K)-cot((beta-phi_a)*K));
    h2 = (vert_B2*sin(beta*K)*cot((beta+vert_phi_2)*K)-vert_B2*cos(beta*K))/
        (cot((beta+phi_b)*K)-cot((beta+vert_phi_2)*K));

    diff = fabs(h1-h2);

    if (diff < diff_stored)
    {
        diff_stored = diff;
        h_pitch = (h1 + h2)/2;
        pitch = beta;
    }
}

//Combine roll and pitch
h1 = h_pitch*cos(roll*K);
h2 = h_roll*cos(pitch*K);
h = (h1 + h2)/2;

printf("\n");
printf("Roll Altitude: %f\n",h_roll);
printf("Roll: %f\n",roll);
printf("Pitch Altitude: %f\n",h_pitch);
printf("Pitch: %f\n",pitch);
printf("Final Altitude: %f\n",h);

gettimeofday(&end,NULL);
seconds = end.tv_sec - start.tv_sec;
useconds = end.tv_usec - start.tv_usec;

```

```
printf(
"Elapsed time for image calcs: %ld seconds, %ld microseconds\n",
seconds,
useconds);
printf("\n");
}
return 0;
}
```